1-1-2011

Authenticated key exchange under bad randomness

Guomin Yang
University of Wollongong, gyang@uow.edu.au

Shanshan Duan
University of California San Diego

Duncan Wong
City University of Hong Kong, dwong@uow.edu.au

Chik How Tan
National University of Singapore

Huaxiong Wang
Nanyang Technological University

Authenticated key exchange under bad randomness

Abstract

We initiate the formal study on authenticated key exchange (AKE) under bad randomness. This could happen when (1) an adversary compromises the randomness source and hence directly controls the randomness of each AKE session; and (2) the randomness repeats in different AKE sessions due to reset attacks.We construct two formal security models, Reset-1 and Reset-2, to capture these two bad randomness situations respectively, and investigate the security of some widely used AKE protocols in these models by showing that they become insecure when the adversary is able to manipulate the randomness. On the positive side, we propose simple but generic methods to make AKE protocols secure in Reset-1 and Reset-2 models. The methods work in a modular way: first, we strengthen a widely used AKE protocol to achieve Reset-2 security, then we show how to transform any Reset-2 secure AKE protocol to a new one which also satisfies Reset-1 security.

Keywords

under, bad, randomness, key, authenticated, exchange

Disciplines

Engineering | Science and Technology Studies

Publication Details

Yang, G., Duan, S., Wong, D., Tan, C. & Wang, H. (2011). Authenticated key exchange under bad randomness. In G. Danezis (Eds.), Financial Cryptography and Data Security Conference: 15th International Conference (pp. 1-23). Germany: Springer.

Authenticated Key Exchange under Bad Randomness*

Guomin Yang[1], Shanshan Duan[2], Duncan S. Wong[3], Chik How Tan[1], and Huaxiong Wang[4]

[1] National University of Singapore
{tslyg,tsltch}@nus.edu.sg
[2] University of California San Diego
shduan@cs.ucsd.edu
[3] City University of Hong Kong
duncan@cityu.edu.hk
[4] Nanyang Technological University
hxwang@ntu.edu.sg

Abstract. We initiate the formal study on authenticated key exchange (AKE) under bad randomness. This could happen when (1) an adversary compromises the randomness source and hence directly controls the randomness of each AKE session; and (2) the randomness repeats in different AKE sessions due to reset attacks. We construct two formal security models, Reset-1 and Reset-2, to capture these two bad randomness situations respectively, and investigate the security of some widely used AKE protocols in these models by showing that they become insecure when the adversary is able to manipulate the randomness. On the positive side, we propose simple but generic methods to make AKE protocols secure in Reset-1 and Reset-2 models. The methods work in a modular way: first, we strengthen a widely used AKE protocol to achieve Reset-2 security, then we show how to transform any Reset-2 secure AKE protocol to a new one which also satisfies Reset-1 security.

Keywords: Authenticated Key Exchange, Resettable Cryptography, Bad Randomness

1 Introduction

An Authenticated Key Exchange (AKE) protocol consists of a tuple of randomized algorithms that enable two parties communicating over an insecure network to establish a common session key. These algorithms consume random coins which are typically generated by pseudo-random number generators (PRNGs). Practical PRNGs, such as ANSI X9.17 PRNG and FIPS 186 PRNG, can generate bit-strings which are computationally indistinguishable from truly random strings provided that the seeds of the PRNGs are fresh and truly random [17]. In practice (e.g. OpenSSL), seeds are formed by collecting random data from an entropy pool, which can be created from a hardware source (e.g., sound/video input, disk drives, etc.) and/or a non-hardware source (e.g., the timing and content of events such as mouse movement, keystroke, network traffic, etc.) [20, 31].

In some practical situations however, the entropy pool could be controlled by an adversary and the seeds may no longer be fresh or truly random. For example, if an adversary has physical access to a hardware source and/or can control the events of a non-hardware source [20], the adversary may be able to manipulate the data in the entropy pool. Also, in some operating systems such as Linux, random data from the entropy pool may be pre-generated and stored in a buffer for later use. If the buffer is not well-protected, an adversary may be able to modify these pre-generated random data.

Adversarial reset of machines could make an AKE protocol reuse the same random coins in different sessions. It has been known as a real threat to some computing devices such as smart cards [13, 7]. Recently, Ristenpart and Yilek [35] showed that the adversarial reset is also a serious security threat to Virtual Machines (VMs). Generally, a system administrator can take snapshots

* A preliminary version of this paper appears in *Financial Cryptography and Data Security 2011*. This is the full version.

of the current system state of a VM from time to time as regular backups. The VM can later be reverted back to a previous state using the snapshots. To perform an adversarial reset, an adversary can first make a system on a VM crash, e.g., via a Denial of Service (DoS) attack. Then when the system administrator reverts the VM back to a "good state", some random coins that have occurred before the machine being reset would be reused after the VM is reverted [35].

We categorize the threats above into "Reset-1" and "Reset-2" attacks, respectively. In Reset-1 attack, the adversary controls the random coins used by the AKE algorithms, while in Reset-2 attack, the adversary can reset a device to make algorithms reuse some random coins. The practical interests of these attacks bring a natural question: would the existing AKE protocols, especially those widely used ones, be still secure under these attacks?

In this paper, we conduct the first formal study on AKE under Reset-1 and Reset-2 attacks. Below are the three aspects to which we contribute.

Security Models. We propose two formal security models for Reset-1 and Reset-2 attacks respectively. We build our models based on the existing Bellare-Rogaway (BR) [8] and Canetti-Krawczyk (CK) [14] security models by providing the adversary additional capabilities. In the Reset-1 model, the adversary directly picks random coins for the AKE participants, while in the Reset-2 model, the adversary does not pick random coins directly but can reset a participant so that the same random coins will be used in multiple AKE sessions. In addition, to capture Kaliski's online *Unknown Key Share* (UKS) attacks [12], our models allow the adversary to register malicious users with public keys of its own choice.

In the Reset-1 model, since the adversary already controls the randomness, if the adversary also learns the long-lived key of either participant, it can trivially compute the session key. Hence the model cannot allow the adversary to corrupt the long-lived key of either participant, and *Forward Secrecy* (FS) cannot be captured. On the other side, the Reset-2 model does not have this restriction. This also indicates that the two models are incomparable and explains the reason why we need two models.

Security of Existing Protocols. Based on the two new models we defined, we show that some well-known AKE protocols (e.g. ISO [24, 14], SIGMA[5] [15, 27], JFK [2], SKEME [28], HMQV [26]), which are proven secure in their original security models will become insecure when the adversary is allowed to manipulate the randomness in the way defined in Reset-1 and/or Reset-2.

Designing New Protocols. We then present techniques to build AKE protocols that are secure in both Reset-1 and Reset-2 models. We present a generic way to efficiently transform a Reset-2 secure AKE protocol to a new one which is secure in both Reset-1 and Reset-2 models. Our idea is to generate good "internal" randomness within the protocol, we do this by applying a pseudo-random function (PRF) to the "raw" randomness at the very beginning of a protocol so that after this treatment, computationally "good" randomness are used in the remaining steps of the protocol. And we prove that this simple (but useful) idea indeed works. However, we remark that some additional requirement on the PRF is needed in order to make the transformed protocol still be Reset-2 secure.

Our transformation provides a "modular approach" to the construction of Reset-1 and Reset-2 secure AKE protocols: (1) we first build a Reset-2 secure AKE protocol Π; (2) then we apply the transformation to Π and get a new protocol Π' which maintains Reset-2 security and in addition to this, it also satisfies Reset-1 security. We illustrate this modular approach by proposing a new SIG-DH based AKE protocol which is transformed from the ISO protocol. The modifications we made are simple and efficient, and can be deployed easily to existing implementations of the protocol.

[5] SIGMA serves as the basis of the signature-based mode of IKE [23] and IKEv2 [25].

Paper Outline. The rest of the paper is organized as follows. In Sec. 2, we review some related work on AKE and resettable cryptography. In Sec. 3, we formalize Reset-1 and Rest-2 security for AKE protocols. In Sec. 4, we show that a collection of widely-used AKE protocols do not satisfy Reset-1 or Reset-2 security. In Sec. 5, we propose a method to generically transform a Reset-2 secure AKE protocol to a new one which is also Reset-1 secure. This is followed by the description of two new AKE protocols that are secure in both Reset-1 and Reset-2 security models. Both of them are derived from some widely-used AKE protocols. The paper is concluded in Sec. 7.

2 Related Work

Authenticated key exchange (AKE) protocols have been extensively studied within the cryptographic community. The first complexity-theoretic treatment of the security notion of AKE is due to Bellare and Rogaway [8]. Their model (referred to as the BR model) and its variants (e.g., [9, 10, 14]) then became the *de facto* standard for analyzing AKE protocols. In [14], Canetti and Krawczyk combined previous work and presented a new security model (referred to as the CK model). They showed that AKE protocols secure in their model can be composed with symmetric key encryption and authentication functions to provide provably secure communication channels. The CK model was used to demonstrate the security of many popular AKE protocols such as the ISO protocol [24, 14], the SIGMA protocol [15, 27], the HMQV protocol [26] and many more. Recently, LaMacchia et al. [29] extended the CK model to a new model (referred to as eCK model). In their model, the adversary is allowed to compromise either the long-live keys or the ephemeral keys (the latter are related to the randomness) of the participants of a protocol session. There have been many discussions on the strength of the two (CK and eCK) models [29, 33, 11]. In [11], Boyd et al. suggested that these two models are incomparable. In this paper, we follow the definitional approach of Canetti and Krawczyk, and we will see later that in fact our Reset-2 model can be considered as Resettable CK model.

None of the existing security models for AKE considers the bad randomness scenarios that we describe in this paper. Although for AKE protocols resilience to the leakage of ephemeral secret key has been studied by Krawczyk [26] and by LaMacchia et al. [29], their work is different from ours: firstly, ephemeral secret key is related but not equivalent to the randomness required by an AKE protocol, in particular, randomness may be required in other parts of the protocol; secondly, in the case of ephemeral secret key leakage, the adversary can only *passively* learn the ephemeral secret key, but not control its value. Another piece of work that is "somehow" related to ours is the work by Aiello et al. [2] in which the JFK (Just Fast Keying) AKE protocol was proposed and discussions about the reuse of Diffie-Hellman (DH) exponents in multiple JFK AKE sessions were made. However, this is different from the Reset-2 scenario we discussed earlier. The reuse of DH exponent is initiatively implemented by a participant of the protocol for reducing the number of costly modular exponentiation operations. But in a reset attack, all the components of the protocol use unfresh/used randomness. To see the difference more clearly, if an AKE protocol (such as JFK) uses a randomized digital signature scheme (such as Digital Signature Standard - DSS [1]), then reusing the same randomness to sign different messages may allow an adversary to derive the secret signing key. However, merely reusing the DH exponent may not cause such a serious consequence.

Resettable Cryptography. Resettable security have been considered for other cryptographic protocols before, such as resettable Zero-Knowledge (rZK) proof [13] and resettable Identification (rID) protocols [7]. Recently, Goyal and Sahai [22] studied the problem of resettably secure two-party and multi-party computation for general functionalities, and in [36], Yilek studied resettably secure public key encryption. Although AKE can be considered as a two-party computation func-

tion, our work is different from that of Goyal and Sahai [22]. We focus on examining and enhancing the existing AKE protocols that have been widely used in the real practice.

HEDGED CRYPTOGRAPHY. Our paper is not the first paper to treat bad randomness for cryptographic operations. The Hedged Cryptography [4, 35] preprocesses randomness together with other inputs (messages, keys, etc.) of a cryptographic operation to provide (pseudo)randomness for the cryptographic operation. In particular, in [35], Ristenpart and Yilek presented the hedged RSA key transport and authenticated Diffie-Hellman key exchange protocols used in TLS without formal security models. Their heading technique is different from our treatment to the randomness presented in Sec. 5 and their hedged protocols cannot provide Reset-1 security.

3 Security Models and Definitions

3.1 AKE Protocol Descriptions

An Authenticated Key Exchange (AKE) protocol consists of two probabilistic polynomial time algorithms: the Long-Lived Key generation algorithm SKG and a protocol execution algorithm P. In this paper, we focus on the public key setting where the algorithm SKG returns a public key and a private key upon each invocation.

PROTOCOL PARTICIPANTS. We initialize a nonempty set $\mathcal{U}$ of parties. Each party $U \in \mathcal{U}$ is named by a unique string, and that string has some fixed length. We use another set $\mathcal{MU}$ to denote malicious parties who are added into the system by an adversary after the initialization phase. Each malicious party $M \in \mathcal{MU}$ is also named by a distinct and fixed-length string which has never been used to name another party inside the system.

LONG-LIVED KEYS. Each party $U \in \mathcal{U}$ holds a public/private key pair $(\mathrm{pk}_U, \mathrm{sk}_U)$ that is generated according to the Long-Lived Key generation algorithm SKG. However, for each party $M \in \mathcal{MU}$, its public key pk_M can be set to any value except that pk_M has never been used as the public key of another party inside the system.

INSTANCES. A party may run many instances concurrently. We denote instance i of party U by Π_U^i. At the time a new instance is created, a unique instance number within the party is chosen, a sequence of random coins are tossed and feeded to that instance, and the instance enters the "ready" state.

PROTOCOL EXECUTION. A protocol execution algorithm is a probabilistic algorithm taking strings to strings. This algorithm determines how instances of the parties behave in response to signals (messages) from their environment. Upon receiving an incoming signal (message) M_{in}, an instance runs the protocol P and generates

$$(M_{\mathrm{out}}, \mathsf{acc}, \mathsf{term}_U^i, \mathsf{sid}_U^i, \mathsf{pid}_U^i, \mathrm{ssk}, St_U^i) \leftarrow \mathrm{P}(1^k, U, \mathrm{pk}_U, \mathrm{sk}_U, St_U^i, M_{\mathrm{in}}).$$

The first component M_{out} corresponds to the responding message, the second component acc denotes the *decision* the instance has made, and the third component term_U^i indicates if the protocol execution has been terminated. A session id (sid_U^i), and partner id (pid_U^i) may be generated during the protocol execution. When the decision is *accept*, the instance holds a session key (ssk) which is to be used by upper layer applications. For all the protocols we analyze in this paper, we assume the state information St_U^i is erased from the memory of U once term_U^i becomes true.

PARTNERSHIP. The partnership between two instances is defined via parter ID (pid) and session ID (sid). The pid names the party with which the instance believes it has just exchanged a key, and the sid is an identifier which uniquely labels the AKE session. We say two instances Π_U^i and Π_V^j are partners if $\mathsf{pid}_U^i = V, \mathsf{pid}_V^j = U$ and $\mathsf{sid}_U^i = \mathsf{sid}_V^j$.

procedure Initialize
For all $U \in \mathcal{U}$
  $(\mathrm{pk}_U, \mathrm{sk}_U) \leftarrow\!\!\$\ \mathsf{SKG}(1^k, U)$; $T_U \leftarrow \emptyset$
Timer $\leftarrow 0$; $b \leftarrow\!\!\$\ \{0,1\}$; $\mathcal{MU} \leftarrow \emptyset$
return $\{\mathrm{pk}_U\}_{U \in \mathcal{U}}$

procedure Register(U, pk)
If $(U \in (\mathcal{U} \cup \mathcal{MU}) \vee \mathrm{pk} \in \{\mathrm{pk}_V\}_{V \in \mathcal{U} \cup \mathcal{MU}})$ then
  return *Invalid*
$\mathcal{MU} \leftarrow \mathcal{MU} \cup \{U\}$
return true

procedure NewInstance(U, i, N)
If $(U \notin \mathcal{U} \vee i \in T_U)$ then return *Invalid*
$T_U \leftarrow T_U \cup \{i\}$; $\mathrm{N}^i_U \leftarrow \mathrm{N}$; $St^i_U \leftarrow (\mathrm{N}^i_U, \mathsf{ready})$
$\mathsf{acc}^i_U \leftarrow \mathsf{false}$; $\mathsf{term}^i_U \leftarrow \mathsf{false}$
$\mathsf{sid}^i_U \leftarrow \perp$; $\mathsf{pid}^i_U \leftarrow \perp$; $\mathrm{ssk}^i_U \leftarrow \perp$
return true

procedure Send(U, i, M_{in})
If $(U \notin \mathcal{U} \vee i \notin T_U \vee \mathsf{term}^i_U)$ then return *Invalid*
$(M_{\mathrm{out}}, \mathsf{acc}, \mathsf{term}^i_U, \mathsf{sid}^i_U, \mathsf{pid}^i_U, \mathrm{ssk}, St^i_U)$
  $\leftarrow \mathrm{P}(1^k, U, \mathrm{pk}_U, \mathrm{sk}_U, St^i_U, M_{\mathrm{in}})$
If $(\mathsf{acc} \wedge$ not $\mathsf{acc}^i_U)$ then
  $\mathrm{ssk}^i_U \leftarrow \mathrm{ssk}$; $\mathsf{acc}^i_U \leftarrow \mathsf{true}$
return $(M_{\mathrm{out}}, \mathsf{acc}, \mathsf{term}^i_U, \mathsf{sid}^i_U, \mathsf{pid}^i_U)$

procedure Reveal(U, i)
If $(U \notin \mathcal{U} \vee i \notin T_U)$ then return *Invalid*
Timer $\leftarrow$ Timer $+ 1$; $\mathsf{Time}[\mathbf{Reveal}, (U, i)] \leftarrow$ Timer
return ssk^i_U

procedure Corrupt(U)
If $U \notin \mathcal{U}$ then return *Invalid*
Timer $\leftarrow$ Timer $+ 1$
$\mathsf{Time}[\mathbf{Corrupt}, U] \leftarrow$ Timer
return sk_U

procedure Test(U^*, i^*)
If $U^* \notin \mathcal{U}$ then return *Invalid*
If (not $\mathsf{acc}^{i^*}_{U^*}$) then return *Invalid*
$K \leftarrow\!\!\$$ KeySpace
If $b = 0$ then return K
Else return $\mathrm{ssk}^{i^*}_{U^*}$

procedure Finalize(b')
$V^* \leftarrow \mathsf{pid}^{i^*}_{U^*}$
If $V^* \notin \mathcal{U}$ then return false
If ($\mathsf{Time}[\mathbf{Corrupt}, U^*]$
  $\vee \mathsf{Time}[\mathbf{Corrupt}, V^*]$)
  return false
If ($\mathsf{Time}[\mathbf{Reveal}, (U^*, i^*)]$)
  return false
If $(\exists i, i \neq i^* \wedge \mathrm{N}^i_{U^*} = \mathrm{N}^{i^*}_{U^*})$
  return false
If $(\exists j^* \in T_{V^*}, \mathsf{pid}^{j^*}_{V^*} = U^*$
  $\wedge \mathsf{sid}^{j^*}_{V^*} = \mathsf{sid}^{i^*}_{U^*})$
  If $(\exists j, j \neq j^* \wedge \mathrm{N}^j_{V^*} = \mathrm{N}^{j^*}_{V^*})$
    return false
  If ($\mathsf{Time}[\mathbf{Reveal}, (V^*, j^*)]$)
    return false
return $(b = b')$

Fig. 1. Game RAKE-1.

3.2 Security Models

We define two security models to capture the two scenarios (namely, Reset-1 and Reset-2) where the randomness of an AKE protocol goes bad. However, we assume that the long-lived keys of all the honest party in the set $\mathcal{U}$ are securely generated using fresh random coins.

Reset-1 Model. In this model, we consider the scenario where the randomness of each instance is completely controlled by the adversary. The formal definition is given in Figure 1 where in total six types of oracle queries are defined to capture the adversarial capabilities. In the following we explain those oracle queries in detail.

Register(U, pk_U) This oracle query allows the adversary A to register a new user U with public key pk_U. Here we only require that neither the user identity U nor the public key pk_U exists in the system. In particular, we do not require the adversary to provide a proof of knowledge on the secret key with regard to pk_U.

NewInstance(U, i, N) This oracle query allows A to initialize a new instance Π^i_U within party U with a binary string N which serves as the random tape of Π^i_U.

Send(U, i, M_{in}) This oracle query invokes instance i of U with message M_{in}. The instance then runs $\text{P}(1^k, U, \text{pk}_U, \text{sk}_U, St_U^i, M_{\text{in}})$ and sends the response back to the adversary. Should Π_U^i terminate or accept will be made available to A. The session id sid_U^i and partner id pid_U^i are also made available to A once they are available.

Reveal(U, i) If oracle Π_U^i has accepted and generated a session key ssk_U^i, then ssk_U^i is returned to the adversary.

Corrupt(U) By making this oracle query, adversary A obtains the long-lived secret key sk_U of party U.

Test(U^*, i^*) By making this oracle query, A selects a challenge instance $\Pi_{U^*}^{i^*}$. If $\Pi_{U^*}^{i^*}$ has accepted, holding a session key $\text{ssk}_{U^*}^{i^*}$, then the following happens. If the coin b, flipped in the **Initialize** phase, is 1, then $\text{ssk}_{U^*}^{i^*}$ is returned to the adversary. If $b = 0$, then a random session key is drawn from the session key space and returned to the adversary. This query is only asked once during the whole game.

The success of an adversary is measured by its ability to distinguish a real session key from a random key in the session key space. However, some oracle queries will render session keys exposed. By issuing these queries the adversary can trivially win the game. To exclude these trivial attacks, we consider the adversary to be successful only if it specifies a *fresh* oracle in the **Test** query.

First of all, the adversary can trivially derive a session key if one of the parties involved in that session is the adversary itself (i.e. one party is created by the adversary via a **Register** query).

The adversary will learn a party's long lived key by making a **Corrupt** query. Since in the Reset-1 model, randomness is completely controlled by the adversary, once a party is corrupted, the adversary is able to derive all state information and session keys ever generated by the party. So there is no security guarantee on session keys of any corrupted party. In other words, we don't consider the notion of *forward secrecy* in the Reset-1 model.

The adversary can certainly learn the value of a session key via a **Reveal** query. In the reset setting, the adversary can also derive a session key by mounting the *reset-and-reply* attack. Specifically, the adversary first activates a protocol execution between instance Π_U^i with random tape N_U, and instance Π_V^j with random tape N_V. Then it activates another instance $\Pi_U^{i'}$ with the same random tape N_U. By replaying messages from Π_V^j, the adversary makes $\text{ssk}_U^{i'} = \text{ssk}_U^i$. In this case, revealing ssk_U^i (or using it in a upper layer application) will automatically render $\text{ssk}_U^{i'}$ insecure, and vice versa. This type of attacks imply that as long as the random tape of one instance Π_U^i is used by another instance $\Pi_U^{i'}$, there is no security guarantee on the session keys generated by these two instances. So when defining the freshness of an instance, we require that its random tape is never used by another instance. Our goal is to design AKE protocols such that reset attacks would not affect the security of session keys generated by those un-reset instances.

Definition 1. *Let $\mathcal{AKE}$ be an AKE protocol. Let A be a Reset-1 adversary against $\mathcal{AKE}$ and k a security parameter. The advantage of A is defined as*

$$\mathbf{Adv}^{\text{rake-1}}_{\mathcal{AKE},A}(k) = \Pr\left[\,\text{RAKE-1}_{\mathcal{AKE},A}(k) \Rightarrow \mathsf{true}\,\right] - 1/2\ .$$

We say $\mathcal{AKE}$ is secure in the Reset-1 model if

1. *in the presence of a benign adversary who faithfully conveys messages, then two partnering instances output the same session key; and*
2. *for any PPT adversary A, $\mathbf{Adv}^{\text{rake-1}}_{\mathcal{AKE},A}(k)$ is negligible.*

Reset-2 Model. In this model, we consider the scenario where the adversary is able to perform reset attacks, but unable to directly set the value of the random coins. The game RAKE-2, described

procedure Initialize
The same as in Game RAKE-1

procedure Register(U, pk)
The same as in Game RAKE-1

procedure NewInstance(U, i, j)
If $(U \notin \mathcal{U} \vee i \in T_U)$ then return *Invalid*
If $j \neq \perp \wedge j \notin T_U$ then return *Invalid*
If $j = \perp$ then $\mathrm{R}_U^i \leftarrow\!\!\$$ RandomCoins
Else $\mathrm{R}_U^i \leftarrow \mathrm{R}_U^j$
$T_U \leftarrow T_U \cup \{i\}$; $St_U^i \leftarrow (\mathrm{R}_U^i, \mathsf{ready})$
$\mathsf{acc}_U^i \leftarrow \mathsf{false}$; $\mathsf{term}_U^i \leftarrow \mathsf{false}$
$\mathsf{sid}_U^i \leftarrow \perp$; $\mathsf{pid}_U^i \leftarrow \perp$; $\mathrm{ssk}_U^i \leftarrow \perp$
return true

procedure Send(U, i, M_{in})
The same as in Game RAKE-1

procedure Reveal(U, i)
The same as in Game RAKE-1

procedure Corrupt(U)
The same as in Game RAKE-1

procedure Test(U^*, i^*)
The same as in Game RAKE-1

procedure Finalize(b')
$V^* \leftarrow \mathsf{pid}_{U^*}^{i^*}$
If $V^* \notin \mathcal{U}$ then return false
If $(\exists i, i \neq i^* \wedge \mathrm{R}_{U^*}^i = \mathrm{R}_{U^*}^{i^*})$
  return false
If $(\mathsf{Time}[\mathbf{Reveal}, (U^*, i^*)])$
  return false
If $(\exists j^* \in T_{V^*}, \mathsf{pid}_{V^*}^{j^*} = U^*$
  $\wedge \mathsf{sid}_{V^*}^{j^*} = \mathsf{sid}_{U^*}^{i^*})$
  If $(\exists j, j \neq j^* \wedge \mathrm{R}_{V^*}^j = \mathrm{R}_{V^*}^{j^*})$
    return false
  If $(\mathsf{Time}[\mathbf{Reveal}, (V^*, j^*)])$
    return false
Else
  If $(\mathsf{Time}[\mathbf{Corrupt}, V^*])$
    return false
return $(b = b')$

Fig. 2. Game RAKE-2.

in Figure 2, is used to define the security of AKE protocols in the Reset-2 setting. The definitions of oracle queries **Register**, **Send**, **Reveal**, **Corrupt** and **Test** are the same as those in game RAKE-1. But differently, when initializing a new user instance Π_U^i via a **NewInstance** query, the adversary does not set the random coins directly. Instead, it can specify another instance Π_U^j that has already been initialized, and instance Π_U^i would use the same random coins that Π_U^j has used. The adversary can also let Π_U^i use fresh random coins by setting $j = \perp$.

This adversarial model enables us to define *forward secrecy*. Recall that forward secrecy requires that compromising two users' long-lived secret keys should not allow the adversary to compromise any already established session key. We say an instance Π_U^i $(U \in \mathcal{U})$ is fs-unfresh in the Reset-2 model if any of the following conditions is true:

1. pid_U^i is created by the adversary via a **Register** query.
2. A reveals the session key of Π_U^i.
3. There exists another instance of U whose random tape is the same as that of Π_U^i (i.e. a reset attack against Π_U^i has occured).
4. Condition 2 is true regarding the partner-oracle of Π_U^i (if it exists).
5. Condition 3 is true regarding the partner-oracle of Π_U^i (if it exists).
6. Π_U^i has no partner instance, and A corrupts pid_U^i.

Otherwise, we say Π_U^i is fs-fresh.

Definition 2. *Let $\mathcal{AKE}$ be an AKE protocol. Let A be a Reset-2 adversary against $\mathcal{AKE}$ and k a security parameter. The advantage of A is defined as*

$$\mathbf{Adv}^{\text{rake-2}}_{\mathcal{AKE},A}(k) = \Pr\left[\,\text{RAKE-2}_{\mathcal{AKE},A}(k) \Rightarrow \mathsf{true}\,\right] - 1/2\ .$$

We say $\mathcal{AKE}$ is secure in the Reset-2 model if

1. *in the presence of a benign adversary who faithfully conveys messages, then two partnering instances output the same session key; and*
2. *for any PPT adversary A,* $\mathbf{Adv}^{\text{rake-2}}_{\mathcal{AKE},A}(k)$ *is negligible.*

Strong Corruption. So far we only consider the so called "weak corruption model". To define strong corruption in our models, we follow the approach of Canetti and Krawczyk [14] and introduce a new query called **RevealState** query.

procedure RevealState(U, i)
If $(U \notin \mathcal{U} \vee i \notin T_U)$ then return *Invalid*
Timer $\leftarrow$ Timer $+ 1$; $\mathsf{Time}[\mathbf{RevealState}, (U, i)] \leftarrow$ Timer
return St^i_U

Now an additional restriction to the adversary A is that A cannot ask the **RevealState** query to the instance $\Pi^{i^*}_{U^*}$ or its partner $\Pi^{j^*}_{V^*}$ (if the latter exists).

procedure Finalize(b')
...
If $(\mathsf{Time}[\mathbf{Reveal}, (U^*, i^*)] \underline{\vee \mathsf{Time}[\mathbf{RevealState}, (U^*, i^*)]})$
return false
...
If $(\mathsf{Time}[\mathbf{Reveal}, (V^*, j^*)] \underline{\vee \mathsf{Time}[\mathbf{RevealState}, (V^*, j^*)]})$
return false
...

It is also worth noting that by adding the **RevealState** query, our Reset-2 model can be considered as Resettable Canneti-Krawczyk model.

4 Security Analysis of Existing Protocols

In this section, we show that several widely used AKE protocols are insecure in our security models. Firstly, it is obvious to see that AKE protocols such as ISO [24], SIGMA [27, 15], JFK [2] and SKEME [28] are insecure in the Reset-1 model since for these protocols, the secrecy of the session key solely relies on the secrecy of the ephemeral secrets. Below we focus on analyzing AKE protocols in the Reset-2 model.

SIG-DH Protocols. A popular approach to design authenticated key exchange protocols is to use the signature-based Diffie-Hellman paradigm. Many popular AKE protocols adopt this approach, such as the ISO protocol (Fig. 3) [24], the SIGMA ("SIGn-and-MAc") protocol [27, 15] and JFK ("Just Fast Keying") [2]. These protocols are all proven secure in the CK model.

The Digital Signature Standard (DSS) [1] is one of the most used options in the real practice (e.g., DSS is recommended in IKEv2 [25]). It is known that for DSS (or any scheme that follows the Fiat-Shamir paradigm [21]), if the same randomness is used to sign two difference messages, then the secret signing key can be recovered with an overwhelming probability.

In either Reset-1 or Reset-2 model, the adversary is able to let an honest party U sign two different messages using the same randomness. The adversary can simply let U and another honest user V perform two AKE sessions, but let U use the same random coins in these two sessions. Since V performs the protocol honestly, with overwhelming probability, the messages signed by U would be different. Then the adversary can retrieve U's signing key.

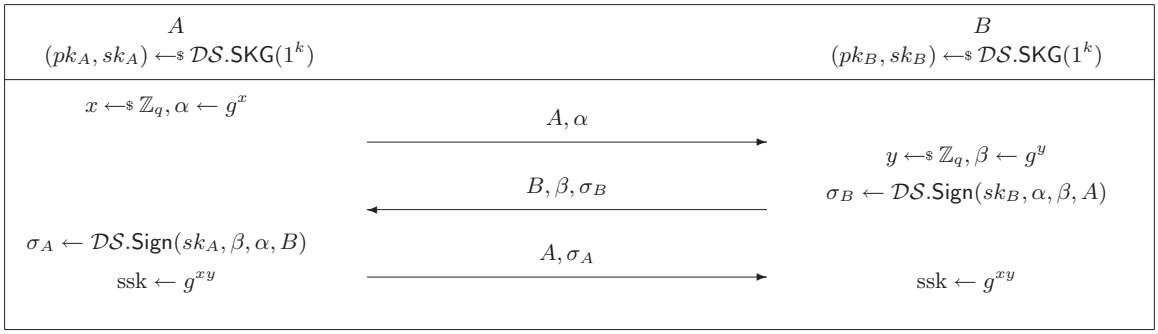

Fig. 3. The ISO Protocol.

HMQV. The HMQV protocol (Fig. 4) is proposed by Krawczyk in [26]. It aims to solve several weaknesses in its predecessor - the MQV protocol [30]. Besides achieving proven security and high efficiency, the HMQV protocol has several extra features, such as resilience to leakage of the DH exponents.

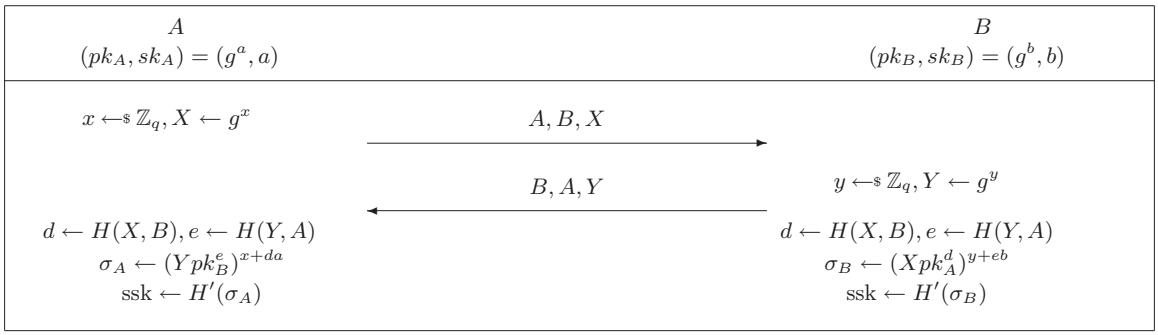

Fig. 4. The HMQV Protocol.

Reset Attack. Below we describe a reset attack originally due to Menezes and Ustaoglu [32]. Assume that the HMQV protocol is implemented in a subgroup G of $\mathbb{Z}_p^*$ with $|\mathrm{G}| = q$ such that $(p-1)/q$ has several small (e.g. less than 2^{40}) pairwise relatively prime factors $t_1, t_2, ..., t_n$ and $\prod_{i=1}^n t_i > q$. The attack works as follows:

1. The adversary first corrupts a party B and obtains B's long lived secret key b.
2. The adversary then activates a new instance of A.
3. After A sends $(A, B, X = g^x)$ to B, the adversary selects a group element $Y \in \mathbb{Z}_p^*$ of order t_1, and sends (B, A, Y) to A.
4. As HMQV does not require A to check the group membership of Y, A will continue the protocol and compute the session key as

$$\sigma_A = (Y pk_B^e)^s = Y^s pk_B^{es} = Y^s (g^s)^{be} = Y^s (X pk_A^d)^{be} \quad \text{and} \quad K = H'(\sigma_A)$$

5. The adversary then issues a Reveal query to get the session key $K = H'(Y^s (X pk_A^d)^{be})$. After that, the adversary iteratively computes $K' = H'(Y^{c_1} (X pk_A^d)^{be})$ for $c_1 = 0, 1, 2, ...$ until $K' = K$ occurs, in which case $c_1 = s = x + da \bmod t_1$.
6. The adversary then resets A and repeats the above process for $t_2, t_3, ..., t_n$.

7. After the adversary obtains $x + da = c_i \bmod t_i$ for $i = 1, 2, ..., n$ (note that since x is the same, d remains unchanged), she can determine $z = x + da \bmod q$ by the Chinese Reminder Theorem.
8. The adversary then corrupts another party C and repeats the above attack again by letting A use the same random coin x. This time the adversary can determine $z' = x + d'a \bmod q$.
9. If $d \neq d'$, which happens with high probability, the adversary can derive A's long lived key as $a = (z - z')/(d - d') \bmod q$.

5 From Reset-2 Security to Reset-1 and Reset-2 Security

In this section, we show that though the Reset-1 and Reset-2 models are incomparable, we can do a simple transformation on a Reset-2 secure AKE protocol to derive a new protocol that is secure in both Reset-1 and Reset-2 models. So to construct a protocol that is secure in both models, we only need to construct one that is secure in the Reset-2 model, and then apply the transformation. Below we first introduce some important tools for our transformation.

Pseudo-Random Function. A family of efficiently computable functions $\mathbb{F} = \{\mathsf{F}_K : \mathsf{D} \to \mathsf{R} | K \in \mathsf{K}\}$ is called a pseudo-random function family, if for any polynomial time algorithm A,

$$\mathbf{Adv}^{\mathrm{prf}}_{\mathbb{F},A}(k) = \Pr\left[A^{\mathsf{F}_K(\cdot)}(1^k) = 1\right] - \Pr\left[A^{\mathsf{RF}(\cdot)}(1^k) = 1\right]$$

is negligible where $K \leftarrow_{\$} \mathsf{K}$, $\mathsf{RF} : \mathsf{D} \to \mathsf{R}$ is a truly random function.

Strong Randomness Extractor [18]. A family of efficiently computable functions $\mathbb{F} = \{\mathsf{F}_K : \mathsf{D} \to \mathsf{R} | K \in \mathsf{K}\}$ is called a strong (m, ϵ)-extractor, if for any random variable X over D that has min-entropy m, if K is chosen uniformly at random from K and R is chosen uniformly at random from R, then the statistical distance between the two distributions $\langle K, \mathsf{F}_K(X)\rangle$ and $\langle K, R\rangle$ is at most ϵ. For our purpose, in fact we only require $\langle K, \mathsf{F}_K(X)\rangle$ and $\langle K, R\rangle$ to be computationally indistinguishable.

The Transformation. Given a protocol $\Pi = (\mathsf{SKG}, \mathrm{P})$ that is secure in the Reset-2 model, and a pseudo-random function family $\mathbb{F} = \{\mathsf{F}_K : \{0,1\}^{\rho(k)} \to \{0,1\}^{\ell(k)} | K \in \{0,1\}^{\delta(k)}\}$ where $\rho(k), \ell(k)$ and $\delta(k)$ are all polynomials of k, and $\ell(k)$ denotes the maximum number of random bits needed by a party in an execution of P, we construct a new protocol $\Pi' = (\mathsf{SKG}', \mathrm{P}')$ as follows:

- $\mathsf{SKG}'(1^k)$: run $\mathsf{SKG}(1^k)$ to generate $(\mathrm{pk}, \mathrm{sk})$, select $K \leftarrow_{\$} \{0,1\}^{\delta(k)}$. Set $\mathrm{pk}' = \mathrm{pk}$ and $\mathrm{sk}' = (\mathrm{sk}, K)$.
- P': get a $\rho(k)$-bit random string r, then compute $r' \leftarrow \mathsf{F}_K(r)$ and run P with random coins r'.

Theorem 1. *If Π is a secure AKE protocol in the Weak-Corruption (Strong-Corruption, resp.) Reset-2 model, and $\mathbb{F}$ is a secure pseudo-random function family, then Π' is a secure AKE protocol in the Weak-Corruption (Strong-Corruption, resp.) Reset-1 model.*

(**Proof Idea**) We prove the Theorem by contradiction. Given an adversary A that breaks Π' in the Reset-1 model, we construct another adversary B breaking Π in the Reset-2 model. Let U^* and V^* denote the parties that A is going to attack, B simulates the Reset-1 game for A by answering queries related to U^* and V^* using its own oracles (notice that B needs to reset U^* (or V^*) if A feeds two instances of U^* (or V^*) with the same "random" coins), and simulates other queries by himself. Since $\mathbb{F}$ is a pseudo-random function family, and A doesn't corrupt U^* or V^*, A wouldn't notice she's in a simulated game, and if she can win the Reset-1 game with a non-negligible advantage, she can win the simulated game (which means B wins the Reset-2 game) with a non-negligible advantage as well.

Proof. The proof is by contradiction, if there exists an adversary A which can break Π' in the Reset-1 model, we construct another adversary B that breaks Π in the Reset-2 model.

First, we define a restricted adversary A_1 in the Reset-1 game as follows. A_1 outputs two different identities U^* and V^* from the set $\mathcal{U}$ after the Initialize phase, and in the Test query, A_1 can only makes a Test query with input (U^*, i^*) which satisfies $\mathsf{pid}_{U^*}^{i^*} = V^*$.

Given an adversary A, we construct A_1 as follows. A_1 randomly chooses from the set $\mathcal{U}$ two identities U^* and V^* after the Initialize Phase, and simulates the Reset-1 game for A by answering all the oracle queries using its own oracles. If A asks a Corrupt query on U^* or V^*, then A_1 outputs a random bit and aborts the game. Let E denote the event that A outputs an instance (I^*, j^*) in the Test query such that $(I^*, \mathsf{pid}_{I^*}^{j^*}) = (U^*, V^*)$. If E does not happen, A_1 outputs a random bit and aborts its execution. Otherwise, if event E occurs, then A_1 makes a Test query also with input (I^*, j^*), and returns the response it gets to A. Finally, when A returns a bit b' and aborts, A_1 also returns b' and aborts.

Since U^* and V^* are randomly selected in the Initialize Phase, we have

$$\begin{aligned}
\mathbf{Adv}^{\text{rake-1}}_{\Pi',A_1}(k) &= \Pr\left[\text{RAKE-1}_{\Pi',A_1}(k) \Rightarrow \mathsf{true} | \mathsf{E}\right]\Pr\left[\mathsf{E}\right] + \Pr\left[\text{RAKE-1}_{\Pi',A_1}(k) \Rightarrow \mathsf{true} | \neg\mathsf{E}\right]\Pr\left[\neg\mathsf{E}\right] - \frac{1}{2} \\
&= \Pr\left[\text{RAKE-1}_{\Pi',A_1}(k) \Rightarrow \mathsf{true} | \mathsf{E}\right]\Pr\left[\mathsf{E}\right] + \frac{1}{2}(1 - \Pr\left[\mathsf{E}\right]) - \frac{1}{2} \\
&= (\Pr\left[\text{RAKE-1}_{\Pi',A_1}(k) \Rightarrow \mathsf{true} | \mathsf{E}\right] - \frac{1}{2})\Pr\left[\mathsf{E}\right] \\
&\geq (\Pr\left[\text{RAKE-1}_{\Pi',A}(k) \Rightarrow \mathsf{true}\right] - \frac{1}{2})\Pr\left[\mathsf{E}\right] \\
&= \frac{1}{n(n-1)}\mathbf{Adv}^{\text{rake-1}}_{\Pi',A}(k)
\end{aligned}$$

GAME $\mathbf{G}^1$. We then modify the Reset-1 game by replacing the function $\mathsf{F}_{K_{U^*}}(\cdot)$ with a truly random function $\mathsf{RF}(\cdot)$ where $\mathsf{F}_{K_{U^*}}(\cdot)$ is the pseudo-random function with key K_{U^*} used by the party U^* in the Reset-1 game. Denote this game by $\mathbf{G}^1$. In the following, we show that A_1 has similar advantages in the Reset-1 game and game $\mathbf{G}^1$. Otherwise, we can construct an adversary D which breaks the pseudo-random function family $\mathbb{F}$.

D has access to an oracle $\mathcal{O}$ which is either a truly random function $\mathsf{RF}(\cdot)$ or a pseudo-random function $\mathsf{F}_K(\cdot)$. D simulates the Reset-1 game by performing all the operations honestly except that D simulates the pseudo-random function $\mathsf{F}_{K_{U^*}}(\cdot)$ of U^* by asking its oracle $\mathcal{O}$. Notice that A_1 never makes a Corrupt query on U^*. Finally, if A_1 wins the game, D outputs 1 and aborts, otherwise, D aborts without any output.

Then we have

$$\begin{aligned}
\mathbf{Adv}^{\text{prf}}_{\mathbb{F},D}(k) &= \Pr\left[D^{\mathsf{F}_K(\cdot)}(1^k) = 1\right] - \Pr\left[D^{\mathsf{RF}(\cdot)}(1^k) = 1\right] \\
&= \Pr\left[A_1 \text{wins the game} | \mathcal{O} = \mathsf{F}_K\right] - \Pr\left[A_1 \text{wins the game} | \mathcal{O} = \mathsf{RF}\right] \\
&= \Pr\left[\text{RAKE-1}_{\Pi',A_1}(k) \Rightarrow \mathsf{true}\right] - \Pr\left[\mathbf{G}^1{}_{\Pi',A_1}(k) \Rightarrow \mathsf{true}\right] \\
&= \mathbf{Adv}^{\text{rake-1}}_{\Pi',A_1}(k) - \mathbf{Adv}^{\mathbf{G}^1}_{\Pi',A_1}(k)
\end{aligned}$$

and

$$\mathbf{Adv}^{\mathbf{G}^1}_{\Pi',A_1}(k) = \mathbf{Adv}^{\text{rake-1}}_{\Pi',A_1}(k) - \mathbf{Adv}^{\text{prf}}_{\mathbb{F},D}(k)$$

GAME $\mathbf{G}^2$. In game $\mathbf{G}^2$, we modify the game $\mathbf{G}^1$ by replacing the function $\mathsf{F}_{K_{V^*}}(\cdot)$ with a truly random function $\mathsf{RF}'(\cdot)$ where $\mathsf{F}_{K_{V^*}}(\cdot)$ is the pseudo-random function with key K_{V^*} used by the party V^* in the game $\mathbf{G}^1$. Then by a similar analysis as in the previous game, we have

$$\mathbf{Adv}^{\mathbf{G}^2}_{\Pi',A_1}(k) = \mathbf{Adv}^{\mathbf{G}^1}_{\Pi',A_1}(k) - \mathbf{Adv}^{\mathrm{prf}}_{\mathbb{F},D}(k)$$

Now given an adversary A_1 against Π' in game $\mathbf{G}^2$, we construct an adversary B against Π in the Reset-2 game. In the Initialization phase for A_1, B gives $\{\mathrm{pk}_U\}_{U\in\mathcal{U}}$ to A_1. After A_1 outputs U^* and V^*, B corrupts all the other $n-2$ parties in the set $\mathcal{U}\backslash\{U^*,V^*\}$ and learns the corresponding long-lived private keys. B also chooses $K_J \leftarrow\!\!\$\ \mathsf{K}$ for all $J \in \mathcal{U}\backslash\{U^*,V^*\}$. B then simulates the game $\mathbf{G}^2$ for A_1 as follows.

When A_1 makes a Register query with input (I, pk_I), B also makes a Register query with the same input.

When A_1 makes a NewInstance query with input (I, j, r),

- If $I \notin \{U^*,V^*\}$, B performs the operations in the NewInstance procedure of game $\mathbf{G}^2$ by himself.
- If $I \in \{U^*,V^*\}$, B first check if A_1 has ever made a NewInstance query with input (I, j', r) where $j' \neq j$. If so, B makes a NewInstance query with input (I, j, j') in the Reset-2 game. Otherwise, B makes a NewInstance query with input $(I, j, \perp)$.

When A_1 makes a Send query with input (I, j, M_{in}),

- If $I \notin \{U^*,V^*\}$, B performs the operations in the Send procedure of game $\mathbf{G}^2$ by himself.
- If $I \in \{U^*,V^*\}$, B makes a Send query in the Reset-2 game also with input (I, j, M_{in}), and returns to A_1 the same response he has received.

B answers the Reveal and RevealState queries in the same way as answering Send queries.

Notice that A_1 never makes a Corrupt query on U^* or V^*, so B answers by himself the Corrupt queries made by A_1.

Suppose A_1 makes a Test query with input (I^*, j^*) (notice that $(I^*, \mathsf{pid}^{j^*}_{I^*}) = (U^*, V^*)$ must be true), B makes a Test query with input (I^*, j^*) in the Reset-2 game, and returns to A_1 the same response he has received.

Finally, when A_1 outputs a bit b' and aborts, B also outputs b' and aborts.

Probability Analysis. We can see that the game simulated by B is identical to game $\mathbf{G}^2$, and if (I^*, j^*) is a fresh session according to game $\mathbf{G}^2$, (I^*, j^*) is also a fresh session in the Reset-2 game. So we have

$$\begin{aligned}
\mathbf{Adv}^{\text{rake-2}}_{\Pi,B}(k) &= \Pr\left[\,\text{RAKE-2}_{\Pi,B}(k) \Rightarrow \mathsf{true}\,\right] - \frac{1}{2} \\
&\geq \Pr\left[\,\mathbf{G}^2_{\Pi',A_1}(k) \Rightarrow \mathsf{true}\,\right] - \frac{1}{2} \\
&= \mathbf{Adv}^{\mathbf{G}^2}_{\Pi',A_1}(k)
\end{aligned}$$

Combining all the results above, we have

$$\mathbf{Adv}^{\text{rake-2}}_{\Pi,B}(k) \geq \frac{1}{n(n-1)}\mathbf{Adv}^{\text{rake-1}}_{\Pi',A}(k) - 2\mathbf{Adv}^{\mathrm{prf}}_{\mathbb{F},D}(k)$$

□

The pseudo-random function (PRF) family $\mathbb{F}$ is the central tool for our transformation. However, the security of a pseudo-random function $\mathsf{F}_K(\cdot)$ relies on the secrecy of the key K. When the key is known to the adversary, then we cannot assume the output of the function is still computationally

indistinguishable from truly random strings. So a problem arises regarding our transformation: the resulting protocol "seems" no longer secure in the Reset-2 model. Recall in the Reset-2 model, the adversary is allowed to corrupt the long-lived key $\mathrm{sk}'_{U^*} = (\mathrm{sk}_{U^*}, K_{U^*})$ of the user U^* that output by the adversary in the Test query, then even given a truly random string r, we cannot guarantee $\mathsf{F}_{K_{U^*}}(r)$ is random from the viewpoint of the adversary who knows K_{U^*}.

Fortunately, this problem can be resolved, but we need an extra requirement on $\mathbb{F}$, that is, we require $\mathbb{F}$ to be a Strong Randomness Extractor (SRE) [18]. In [16], Chevassut et al. showed that those very strong (i.e. the adversary has very small winning advantage) pseudo-random function families are also good strong randomness extractors. For real implementation, the HMAC function [5], which is widely used in the real practice (e.g., TLS and IKE), is a good candidate for our purpose [3, 19, 34].

Theorem 2. *If Π is a secure AKE protocol in the Weak-Corruption (Strong-Corruption, resp.) Reset-2 model, and $\mathbb{F}$ is a pseudo-random function family and a strong randomness extractor, then Π' is secure in the Weak-Corruption (Strong-Corruption, resp.) Reset-2 model.*

Proof. The proof is also by contradiction, if there exists an adversary A which can break Π' ($\mathbb{F}$ is assumed to be a pseudo-random function family and a strong randomness extractor) in the Reset-2 model, we construct another adversary B that breaks Π in the Reset-2 model.

We consider two cases: (1) the instance output by A in the Test query has a partner instance, and (2) the instance output by A in the Test query has no partner instance.

Case 1: the instance (U^*, i^*) output by A in the Test query has a partner instance (V^*, j^*).

Similar to the proof of Theorem 1, we first define a restricted adversary A_1, which outputs two numbers ℓ and ℓ' after the Initialize phase such that the Test session is between the ℓ-th and ℓ'-th instances. Then similar to the proof of the Theorem 1, given an adversary A in the Reset-2 model which makes at most q_I NewInstance queries, we can build A_1 such that

$$\mathbf{Adv}^{\text{rake-2}}_{\Pi',A_1}(k) \geq \frac{1}{q_I(q_I-1)}\mathbf{Adv}^{\text{rake-2}}_{\Pi',A}(k).$$

Game $\tilde{\mathbf{G}}^1$. We then modify the Reset-2 game for A_1 to a new game $\tilde{\mathbf{G}}^1$ such that in the ℓ-th instance (or (U^*, i^*)), the output of the function $\mathsf{F}_{K_{U^*}}(\cdot)$ is replaced with a random string, and in the ℓ'-th instance (or (V^*, j^*)), the output of $\mathsf{F}_{K_{V^*}}(\cdot)$ is replaced with another random string. Then due to that $\mathbb{F}$ is a strong randomness extractor, we have

$$\mathbf{Adv}^{\tilde{\mathbf{G}}^1}_{\Pi',A_1}(k) \geq \mathbf{Adv}^{\text{rake-2}}_{\Pi',A_1}(k) - 2\epsilon_{sre}.$$

Now given an adversary A_1 against Π' in game $\tilde{\mathbf{G}}^1$, we construct an adversary B against Π in the Reset-2 game. B corrupts all the parties in $\mathcal{U}$, selects $K_J \leftarrow_{\$} \mathsf{K}$ for all $J \in \mathcal{U}$, and simulates the game $\tilde{\mathbf{G}}^1$ for A_1.

B answers all the queries made by A_1 except that for all the queries related to the ℓ-th instance (or (U^*, i^*)) and the ℓ'-th instance (or (V^*, j^*)), B relays these queries to its own challenger, and returns to A_1 the responses he receives. Finally, when A_1 outputs a bit b' and aborts, B outputs the same b' and aborts.

The game $\tilde{\mathbf{G}}^1$ simulated by B is perfect, and if A_1 wins the game $\tilde{\mathbf{G}}^1$, B wins the Reset-2 game. So we have

$$\begin{aligned}\mathbf{Adv}^{\text{rake-2}}_{\Pi,B}(k) &= \Pr\left[\,\text{RAKE-2}_{\Pi,B}(k) \Rightarrow \mathsf{true}\,\right] - \frac{1}{2} \\ &\geq \Pr\left[\,\tilde{\mathbf{G}}^1_{\Pi',A_1}(k) \Rightarrow \mathsf{true}\,\right] - \frac{1}{2} \\ &= \mathbf{Adv}^{\tilde{\mathbf{G}}^1}_{\Pi',A_1}(k)\end{aligned}$$

Combining all the results in Case 1, we have

$$\mathbf{Adv}^{\text{rake-2}}_{\Pi,B}(k) \geq \frac{1}{q_I(q_I-1)}\mathbf{Adv}^{\text{rake-2}}_{\Pi',A}(k) - 2\epsilon_{sre}.$$

Case 2: the instance (U^*, i^*) output by A in the Test query has no partner instance.

Similar to Case 1, we first define a restricted adversary A_1 that outputs an integer ℓ and an identity V^* after the Initialize phase such that A_1 outputs the ℓ-th instance (denoted by (U^*, i^*)) in the Test query and $\mathsf{pid}^{i^*}_{U^*} = V^*$. Then similar to the previous proofs, given an adversary A in the Reset-2 game which makes at most q_I NewInstance queries, we can build A_1 such that

$$\mathbf{Adv}^{\text{rake-2}}_{\Pi',A_1}(k) \geq \frac{1}{nq_I}\mathbf{Adv}^{\text{rake-2}}_{\Pi',A}(k).$$

Game $\bar{\mathbf{G}}^1$. In game $\bar{\mathbf{G}}^1$, we modify the Reset-2 game for A_1 such that in the ℓ-th instance (or (U^*, i^*)), the output of the function $\mathsf{F}_{K_{U^*}}(\cdot)$ is replace with a random string. Then similar to the analysis in Case 1, we have

$$\mathbf{Adv}^{\bar{\mathbf{G}}^1}_{\Pi',A_1}(k) \geq \mathbf{Adv}^{\text{rake-2}}_{\Pi',A_1}(k) - \epsilon_{sre}.$$

Game $\bar{\mathbf{G}}^2$. In game $\bar{\mathbf{G}}^2$, we modify game $\bar{\mathbf{G}}^1$ such that we replace the function $\mathsf{F}_{K_{V^*}}(\cdot)$ with a truly random function. Then following a similar analysis as in the proof of Theorem 1, we have

$$\mathbf{Adv}^{\bar{\mathbf{G}}^2}_{\Pi',A_1}(k) \geq \mathbf{Adv}^{\bar{\mathbf{G}}^1}_{\Pi',A_1}(k) - \mathbf{Adv}^{\text{prf}}_{\mathbb{F},D}(k).$$

Given an adversary A_1 against protocol Π' in game $\bar{\mathbf{G}}^2$, we construct another adversary B against protocol Π in the Reset-2 game as follows.

B corrupts all the parties in the set $\mathcal{U}\backslash\{V^*\}$ and learns the corresponding long-lived private keys. B also chooses $K_J \leftarrow_{\$} \mathsf{K}$ for all $J \in \mathcal{U}\backslash\{V^*\}$. B answers by himself all the queries made by A except the following:

- For all the queries that are related to the ℓ-th instance (including the Test query), B relays these queries to its own challenger, and returns to A_1 the responses he receives.
- For all the queries that are related to the party V^*, B also relays the queries to its own challenger, returns to A_1 the responses he receives. Notice that A_1 never makes a Corrupt query on V^*.

Finally, when A_1 outputs a bit b' and aborts, B also outputs b' and aborts. Then we have

$$\begin{aligned}\mathbf{Adv}^{\text{rake-2}}_{\Pi,B}(k) &= \Pr\left[\,\text{RAKE-2}_{\Pi,B}(k) \Rightarrow \mathsf{true}\,\right] - \frac{1}{2} \\ &\geq \Pr\left[\,\bar{\mathbf{G}}^2_{\Pi',A_1}(k) \Rightarrow \mathsf{true}\,\right] - \frac{1}{2} \\ &= \mathbf{Adv}^{\bar{\mathbf{G}}^2}_{\Pi',A_1}(k)\end{aligned}$$

Combining all the results in Case 2, we have

$$\mathbf{Adv}^{\text{rake-2}}_{\Pi,B}(k) \geq \frac{1}{nq_I}\mathbf{Adv}^{\text{rake-2}}_{\Pi',A}(k) - \epsilon_{sre} - \mathbf{Adv}^{\text{prf}}_{\mathbb{F},D}(k).$$

So in conclusion, if we combine the results in both Case 1 and Case 2, we have

$$\mathbf{Adv}^{\text{rake-2}}_{\Pi',A}(k) \leq \mathsf{max}(q_I(q_I-1)(\mathbf{Adv}^{\text{rake-2}}_{\Pi,B}(k) + 2\epsilon_{sre}), nq_I(\mathbf{Adv}^{\text{rake-2}}_{\Pi,B}(k) + \epsilon_{sre} + \mathbf{Adv}^{\text{prf}}_{\mathbb{F},D}(k)))$$

□

6 New SIG-DH and PKE-DH Protocols

In this section, we modify the ISO protocol and the SKEME protocol to obtain new SIG-DH and PKE-DH protocols that are secure in both Reset-1 and Reset-2 models.

6.1 A New SIG-DH Protocol

In Sec. 4, we showed that the ISO protocol is insecure in either reset model. In the following we present a slightly modified protocol (denoted by ISO-R2) and prove its Reset-2 security. Then by applying our transformation in the previous section, we can obtain a new protocol that is secure in both Reset-1 and Reset-2 settings.

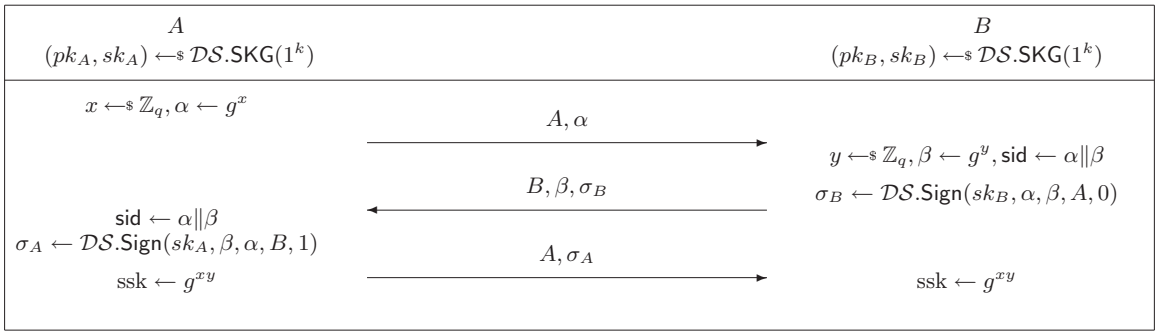

Fig. 5. The ISO-R2 Protocol.

Modification 1. The ISO-R2 protocol (Fig. 5) is similar to the original ISO protocol. One difference is that a role indicator ('0' for responder and '1' for initiator) is added into the message signed by each party. Our modification is to prevent the following interleaving attack when the session id is defined as the concatenation of the random group elements sent by the initiator and the responder.

1 The adversary M corrupts A, and then activates a new session between A (Initiator) and B (Responder).

1' Upon receiving the message (A, α) from A, the adversary activates another session between B (Initiator) and A (Responder)

2' Upon receiving the message (B, β) from B, the adversary sends back $(A, \alpha, \mathcal{DS}.\mathsf{Sign}(sk_A, \beta, \alpha, B))$ to B.

3' B then responses to the adversary M with a message $(B, \mathcal{DS}.\mathsf{Sign}(sk_B, \alpha, \beta, A))$, and accepts the session with $\mathsf{sid} = \beta\|\alpha$.

2 The adversary then sends $(B, \beta, \mathcal{DS}.\mathsf{Sign}(sk_B, \alpha, \beta, A))$ to A.
3 A then responses with $(A, \mathcal{DS}.\mathsf{Sign}(sk_A, \beta, \alpha, B))$ and accepts the session with $\mathsf{sid}' = \alpha \| \beta$.

We can see that A and B would agree with the same session key, but under different session ids. There are several ways to resolve the problem: (1) use an explicit session id (this approach was adopted by Canetti and Krawczyk in [14]) instead of defining the session id as the concatenation of the messages exchanged between the two parties, (2) define the session id as (self-message$\|$peer-message) (in this way, two matching instances generate different session ids) (3) add a role indicator.

The first approach is simple, but it leaves the question of how to concretely define session id. The second way doesn't match our security models in which we assume two partners of a session should have the same session id.

Modification 2. Also, in order to prevent the attack we have described in Sec. 4, we require the underlying digital signature scheme $\mathcal{DS}$ to be deterministic.

Deterministic Digital Signature. We say a digital signature scheme $\mathcal{DS} = (\mathcal{DS}.\mathsf{SKG}, \mathcal{DS}.\mathsf{Sign}, \mathcal{DS}.\mathsf{Vf})$ is deterministic if the signing algorithm $\mathcal{DS}.\mathsf{Sign}$ is deterministic. We can always transform a randomized digital signature scheme to a deterministic one via the following (folklore) trick: the signing key is expanded to include a key K' which is chosen uniformly at random from the key space of a pseudo-random function family $\mathbb{F}'$. To sign a message m, we first compute random coins $r = \mathsf{F}'_{K'}(m)$, and then invoke the (randomized) signing algorithm $\mathcal{DS}.\mathsf{Sign}$ with random coins r.

We say $\mathcal{DS}$ is existentially unforgeable under adaptive chosen message attacks (uf-cma), if for any polynomial time algorithm $\mathcal{F}$,

$$\mathbf{Adv}^{\text{uf-cma}}_{\mathcal{DS},F}(k) = \Pr\left[\begin{array}{l}(pk, sk) \leftarrow \mathcal{DS}.\mathsf{SKG}(1^k), \quad (m^*, \sigma^*) \leftarrow F^{\mathcal{DS}.\mathsf{Sign}(sk,\cdot)}(pk) : \\ \mathcal{DS}.\mathsf{Vf}(pk, m^*, \sigma^*) = 1 \wedge \ F \text{ has never queried } \mathcal{DS}.\mathsf{Sign}(sk, m^*)\end{array}\right]$$

is negligible.

Security Analysis. In the following, we prove that the ISO-R2 protocol is secure in the Reset-2 model under the Decisional Diffie-Hellman (DDH) assumption.

Decisional Diffie-Hellman (DDH) Assumption: The DDH assumption says for any polynomial time algorithm A,

$$\mathbf{Adv}^{\text{DDH}}_{A}(k) = \Pr\left[A(1^k, g, g^a, g^b, Z) = 1 | Z = g^{ab}\right] - \Pr\left[A(1^k, g, g^a, g^b, Z) = 1 | Z = g^r\right]$$

is negligible where a, b, r are randomly selected from $\mathbb{Z}_q$.

Theorem 3. *The ISO-R2 protocol is secure in the Strong Corruption Reset-2 model if $\mathcal{DS}$ is a* uf-cma *secure deterministic digital signature scheme, and the DDH assumption holds in the underlying group.*

Proof. We first define a restricted adversary A_1 such that in the Test query A_1 outputs a instance (U^*, i^*) which has a partner instance (V^*, j^*).

Given an adversary A against the ISO-R2 protocol in the Reset-2 model, we build A_1 as follows. A_1 answers all the queries made by A using its own oracles, if A outputs an instance (U^*, i^*) that has no partner instance, then A_1 aborts without any output. Otherwise, A_1 also outputs (U^*, i^*) in the Test query, and returns to A the response it receives. When A outputs a bit b' and aborts, A_1 also outputs b' and aborts.

We say a Forge event occurs if in the game, the adversary A generates a message signature pair (m^*, σ^*) such that

– There exists an party $I \in \mathcal{U}$ such that $\mathsf{true} \leftarrow \mathcal{DS}.\mathsf{Vf}(\mathrm{pk}_I, m^*, \sigma^*)$, and
– Party I is not corrupted at the time A generates (m^*, σ^*), and
– The party I has never generated a signature on message m^*.

Let E denote the event that A outputs in the Test query an instance (U^*, i^*) that has no partner instance. If E happens, then a Forge event also happens. On the other hand, if E does not happen, then A_1 and A are the same. So we have

$$\mathbf{Adv}^{\text{rake-2}}_{ISO-R2,A}(k) - \mathbf{Adv}^{\text{rake-2}}_{ISO-R2,A_1}(k) \leq \Pr[\mathsf{E}] \leq \Pr[\mathsf{Forge}]$$

In the following we prove that event Forge happens only with negligible probability. Otherwise, we can break the uf-cma security of the deterministic digital signature scheme.

Given an adversary A in the game G^1, we construct a signature forger F as follows: F is given a public key pk where $(pk, sk) \leftarrow \mathcal{DS}.\mathsf{SKG}(1^k)$, and has access to a signing oracle $\mathcal{DS}.\mathsf{Sign}(sk, \cdot)$. F randomly selects a party $U \in \mathcal{U}$, and assigns $\mathrm{pk}_U = pk$. F then generates all the long-lived keys for all the parties in the set $\mathcal{U} \backslash \{U\}$.

F simulates the game G^1 for A. If in the simulation, a Forge event happens and $I = U$, then F outputs the forgery by A and aborts. Then we have

$$\mathbf{Adv}^{\text{uf-cma}}_{\mathcal{DS},F}(k) \geq \frac{1}{n}\Pr[\mathsf{Forge}].$$

So we have

$$\mathbf{Adv}^{\text{rake-2}}_{ISO-R2,A_1}(k) \geq \mathbf{Adv}^{\text{rake-2}}_{ISO-R2,A}(k) - n\mathbf{Adv}^{\text{uf-cma}}_{\mathcal{DS},F}(k).$$

Given adversary A_1 with advantage $\mathbf{Adv}^{\text{rake-2}}_{ISO-R2,A_1}(k)$, we define another restricted adversary A_2 which outputs two integers ℓ and ℓ' after the Initialize phase. A_2 guesses that the Test session output by A_1 is between the ℓ-th and ℓ'-th instances. Then similar to the proof of Theorem 2, given an adversary A_1 as above, we can build an A_2 such that

$$\mathbf{Adv}^{\text{rake-2}}_{ISO-R2,A_2}(k) \geq \frac{1}{q_I(q_I - 1)}\mathbf{Adv}^{\text{rake-2}}_{ISO-R2,A_1}(k).$$

Game G^1. We then modify the Reset-2 game to a new game G^1 as follows. In game G^1, the simulator picks a random key (i.e., a random element in the underlying group), and sets it as the session key of the ℓ-th and the ℓ'-th instances.

In the following, we show that A_2 has similar advantages in the Reset-2 game and game G^1. Otherwise, we can construct an adversary D which breaks the DDH assumption.

D is given a tuple $\{g, X = g^a, Y = g^b, Z\}$ and D's goal is to guess whether $Z = g^{ab}$ or Z is a random group element. D honestly simulates the Reset-2 game for A_2 except that D simulates the ℓ-th instance by setting the ephemeral public key as X, and simulates the ℓ'-th instance by setting the ephemeral public key as Y. D also sets Z as the session key of the ℓ-th and the ℓ'-th instances. If A_1 wins the game, D outputs 1 and aborts. Otherwise, D aborts without any output. Then we have

$$\begin{aligned}
\mathbf{Adv}^{\text{DDH}}_{D}(k) &= \Pr\left[A_2 \text{ wins the game} | Z = g^{ab}\right] - \Pr[A_2 \text{ wins the game} | Z = g^r] \\
&= \Pr[\text{RAKE-2}_{ISO-R2,A_2}(k) \Rightarrow \mathsf{true}] - \Pr\left[G^1_{ISO-R2,A_2}(k) \Rightarrow \mathsf{true}\right] \\
&= \mathbf{Adv}^{\text{rake-2}}_{ISO-R2,A_2}(k) - \mathbf{Adv}^{G^1}_{ISO-R2,A_2}(k)
\end{aligned}$$

Finally, in game G^1, the adversary has no advantage in winning the game (i.e., $\mathbf{Adv}^{G^1}_{ISO-R2,A_2}(k) = 0$). So combing all the results above, we have

$$\mathbf{Adv}^{\text{rake-2}}_{ISO-R2,A}(k) \leq n\mathbf{Adv}^{\text{uf-cma}}_{\mathcal{DS},F}(k) + q_I(q_I-1)\mathbf{Adv}^{\text{DDH}}_{D}(k).$$

□

A SIG-DH Protocol Secure in Both Reset Models. Given the Reset-2 secure ISO-R2 protocol, we can apply the transformation in Sec. 5 to obtain a new protocol (denoted by ISO-R) that is secure in both Reset-1 and Reset-2 models.

Corollary 1. *The ISO-R protocol in Fig. 6 is secure in Strong Corruption Reset-1 and Reset-2 models if $\mathcal{DS}$ is a* uf-cma *deterministic digital signature scheme, $\mathbb{F}$ is a pseudo-random function family and a strong randomness extractor, and the DDH assumption holds in the underlying group.*

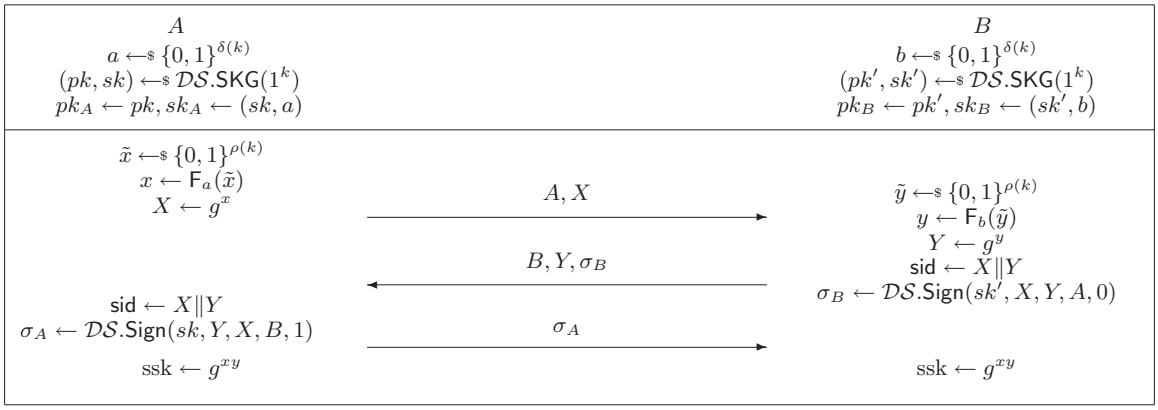

Fig. 6. The ISO-R Protocol.

6.2 A New PKE-DH Protocol

SKEME. The SKEME protocol, proposed by Krawczyk in [28], is a typical PKE-DH protocol. SKEME serves as the basis of the public-key encryption mode of IKE [23]. The SKEME protocol, which uses public key encryption together with message authentication code, is described in Fig. 7.

Public Key Encryption. A public key encryption scheme $\mathcal{PKE}$ consists of three algorithms. The key generation algorithm $\mathcal{PKE}.\mathsf{SKG}(1^k)$ takes a security parameter as input and outputs a public/private key pair $(\mathrm{pk}, \mathrm{sk})$. The encryption algorithm $\mathcal{PKE}.\mathsf{Enc}(\mathrm{pk}, m)$ takes the public key and a message as input, and outputs a ciphertext c. The decryption algorithm $\mathcal{PKE}.\mathsf{Dec}(\mathrm{sk}, c)$ takes the private key and a ciphertext as input, and outputs m or $\perp$ (which indicates decryption failure).

A public key encryption scheme $\mathcal{PKE}$ is secure under adaptive chosen ciphertext attacks if for any PPT adversary $A = (A_1, A_2)$,

$$\mathbf{Adv}^{\text{cca}}_{\mathcal{PKE},A}(k) \stackrel{\text{def}}{=} \Pr\left[\begin{array}{l}(\mathrm{pk},\mathrm{sk}) \leftarrow \mathcal{PKE}.\mathsf{SKG}(1^k), (x_0, x_1, \delta) \leftarrow A_1^{\mathcal{PKE}.\mathsf{Dec}(\mathrm{sk},\cdot)}(\mathrm{pk}), b \leftarrow\!\!\$ \{0,1\}, \\ y \leftarrow \mathcal{PKE}.\mathsf{Enc}(\mathrm{pk}, x_b), b' \leftarrow A_2^{\mathcal{PKE}.\mathsf{Dec}(\mathrm{sk},\cdot)}(\mathrm{pk}, x_0, x_1, \delta, y) : b' = b\end{array}\right] - \frac{1}{2}$$

is negligible, where $|x_0| = |x_1|$, and A_2 is not allowed to make a decryption query with input y.

Message Authentication Code. A message authentication code scheme $\mathcal{MAC}$ with key space $\mathcal{K}$ consists of two algorithms: a message authentication algorithm $\mathsf{MAC}(K, m)$ which takes a key $K \in \mathcal{K}$ and a message m as input and returns an authentication tag τ; and a verification algorithm $\mathsf{MAV}(K, m, \tau)$ which takes a key $K \in \mathcal{K}$, a message m, and an authentication tag τ as input, and returns either 1 or 0.

A message authentication code scheme $\mathcal{MAC}$ is secure under adaptive chosen message attacks, if for any polynomial time algorithm F,

$$\mathbf{Adv}^{\mathrm{cma}}_{\mathcal{MAC},F}(k) = \Pr\left[\begin{array}{l} K \leftarrow\!\!\$\, \mathcal{K}, \quad (m^*, \tau^*) \leftarrow F^{\mathsf{MAC}(K,\cdot)}(1^k) : \\ \mathsf{MAV}(K, m^*, \tau^*) = 1 \wedge \ F \text{ has never queried } \mathsf{MAC}(K, m^*) \end{array}\right]$$

is negligible.

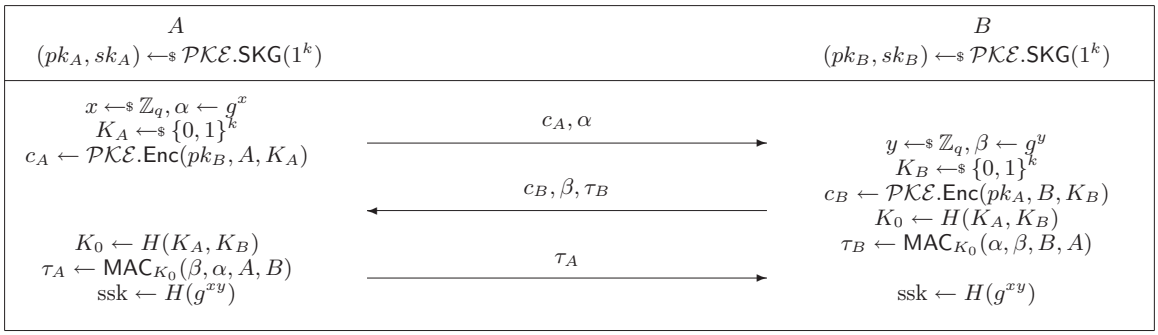

Fig. 7. The SKEME Protocol.

It is worth noticing that the SKEME protocol is insecure in the Strong Corruption model where the adversary can make **RevealState** queries, as shown below.

- The adversary first activates user A to start a new session with user B.
- The adversary relays the message c_A, α from A to B.
- Upon receiving the message c_B, β, τ_B from B, the adversary makes a **RevealState** query to B and obtains the key K_0.
- The adversary generates $\beta' \leftarrow g^{y'}$ where y' is chosen by the adversary, $\tau'_B \leftarrow \mathsf{MAC}_{K_0}(\alpha, \beta', B, A)$, and sends a message (c_B, β', τ'_B) to user A.
- Since the authentication tag τ'_B is generated using the correct key K_0, user A would accept the session and output the session key $H(g^{xy'})$.

SKEME Variant. We present a modified version of the SKEME protocol and show its security in the Weak Corruption Reset-2 model. Then by applying the transformation in Sec. 5, we obtain a new protocol which is also secure in the Weak Corruption Reset-1 model.

Theorem 4. *The PKEDH-R2 protocol is secure in the Weak Corruption Reset-2 model if $\mathcal{PKE}$ is secure under adaptive chosen ciphertext attacks, $\mathcal{MAC}$ is secure under adaptive chosen message attacks, and the DDH assumption holds in the underlying group.*

Proof (Sketch.). Similar to the proof of Theorem 3, we show that when an adversary outputs an instance (U^*, i^*) in the Test query, then (U^*, i^*) must have a partner instance (V^*, j^*), otherwise we can break the public key encryption scheme $\mathcal{PKE}$ or the message authentication code $\mathcal{MAC}$.

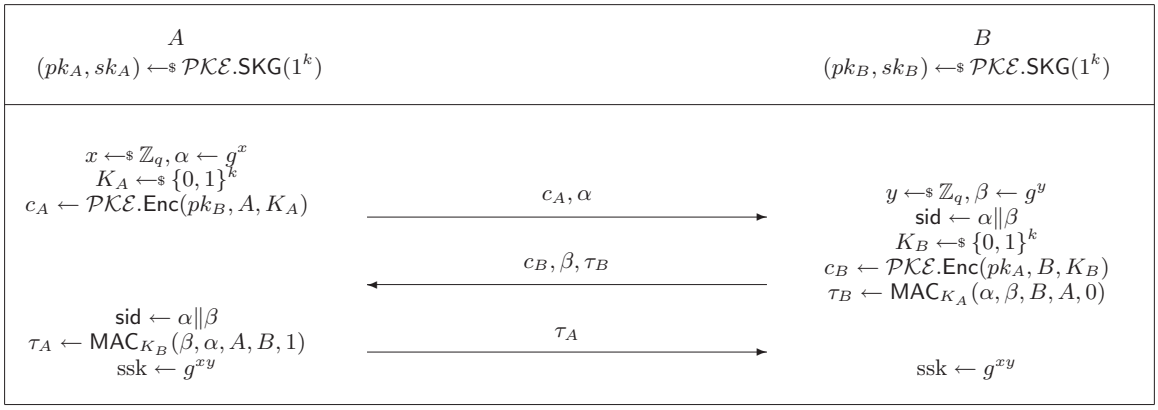

Fig. 8. The PKEDH-R2 Protocol.

The proof follows the lines in [6]. We first define an Encryption Aided Forger F as follows. Let $(pk, sk) \leftarrow \mathcal{PKE}.\mathsf{SKG}(1^k)$, and $c^* \leftarrow \mathcal{PKE}.\mathsf{Enc}(pk, S, N^*)$ where S is a string chosen by F, and N^* is randomly selected from the key space of $\mathcal{MAC}$ and unknown to F. F is given pk, c^*, and has access to two oracles: a decryption oracle $\mathcal{O}_{sk}(\cdot)$ which decrypts ciphertexts different from c^*, and an $\mathcal{O}_{N^*}(\cdot)$ oracle which on input m returns $\mathsf{MAC}_{N^*}(m)$. F's goal is to output $m^*, \mathsf{MAC}_{N^*}(m^*)$ where F has never queried the oracle $\mathcal{O}_{N^*}(\cdot)$ on message m^*.

In the Reset-2 game, if the instance (U^*, i^*) output by the adversary A in the Test query has no partner instance (denote this event by E), then we can construct an Encryption Aided Forger F.

F first randomly selects two parties U^*, V^* from $\mathcal{U}$, and generates all the long-lived keys for other parties in the set $\mathcal{U} \backslash \{V^*\}$. Let q_I denotes the maximum number of NewInstance queries made by A. F also selects an integer $\ell \leftarrow\!\!\$\ [1, q_I]$. F then asks its challenger with input U^*, and gets back the challenge $pk, c^* = \mathcal{PKE}.\mathsf{Enc}(pk, U^*, N^*)$. F then sets $\mathrm{pk}_{V^*} = pk$, and simulates the Reset-2 game for A honestly except that

- If the adversary does not make a Test query with an instance of U^*, F aborts without any output,
- If $\mathsf{pid}^{i^*}_{U^*} \neq V^*$, then F aborts without any output,
- If (U^*, i^*) is not the ℓ-th instance, then F aborts without any output,
- If the adversary makes a Corrupt query with input V^*, F aborts without any output,
- In the ℓ-th instance, F sets $(c_{U^*} = c^*)$, generates the ephemeral DH public and private key pair for (U^*, i^*), uses sk_{U^*} to get $N \leftarrow \mathcal{PKE}.\mathsf{Dec}(\mathrm{sk}_{U^*}, c_{V^*})$, and then generates the tag τ_{U^*} honestly using N.
- If the adversary sends a message $(c, ...)$ to V^* where $c \neq c^*$, F makes a query to its decryption oracle $\mathcal{O}_{sk}(\cdot)$ on input c, and proceeds as usual after getting back the response from $\mathcal{O}_{sk}(\cdot)$.
- If the adversary sends a message $(c^*, ...)$ to V^*, F queries its oracle $\mathcal{O}_{N^*}(\cdot)$ to generate the response tag.
- When A sends the MAC tag to the ℓ-th instance, F outputs the MAC tag and the corresponding message as his forgery and aborts.

Then we have

$$\Pr[F \text{ succeeds}] \geq \frac{1}{n(n-1)q_I} \Pr[\mathsf{E}]$$

Let $\varepsilon = \Pr[F \text{ succeeds}]$. Now given an Encryption Aided Forger F, we construct another adversary D against the public key encryption scheme $\mathcal{PKE}$ in the IND-CCA game. D is given

a public key pk and has access to a decryption oracle. D runs F as follows. When F asks for a challenge with input S, D randomly selects two numbers N_0 and N_1, and asks its challenger with input $S\|N_0$ and $S\|N_1$. After getting back the challenge c^*, D sets pk, c^* as F's challenge. When F makes a decryption query with a ciphertext $c \neq c^*$, D makes a decryption query also with input c. When F makes an MAC query on message m, D returns $\mathsf{MAC}_{N_0}(m)$ to F. Finally, if F successfully makes a forgery $\mathsf{MAC}_{N_0}(m^*)$, then D outputs 0, indicating c^* is an encryption of $S\|N_0$. Otherwise, if F fails to produce a forgery, then D outputs 1, indicating c^* is an encryption of $S\|N_1$. Then we have

$$\begin{aligned}
\mathbf{Adv}^{\mathrm{cca}}_{\mathcal{PKE},D}(k) &= \Pr[\,D \text{ outputs } 0|b=0\,]\Pr[\,b=0\,] + \Pr[\,D \text{ outputs } 1|b=1\,]\Pr[\,b=1\,] - \frac{1}{2} \\
&= \frac{1}{2}\Pr[\,F \text{ succeeds } |b=0\,] + \frac{1}{2}(1 - \Pr[\,F \text{ succeeds } |b=1\,]) - \frac{1}{2} \\
&= \frac{1}{2}(\Pr[\,F \text{ succeeds } |b=0\,] - \Pr[\,F \text{ succeeds } |b=1\,]) \\
&= \frac{1}{2}(\varepsilon - \mathbf{Adv}^{\mathrm{cma}}_{\mathcal{MAC},F}(k))
\end{aligned}$$

The last equality comes from the fact that when $b = 0$, then F is in the Encryption Aided Forger game, and when $b = 1$, then c^* is independent of N_0, and F is in the normal chosen message attack game. Combining all the results, we have

$$\Pr[\,\mathsf{E}\,] \leq n(n-1)q_I(2\mathbf{Adv}^{\mathrm{cca}}_{\mathcal{PKE},D}(k) + \mathbf{Adv}^{\mathrm{cma}}_{\mathcal{MAC},F}(k)).$$

We omit the remaining of the proof, since it is the same as that of Theorem 3. □

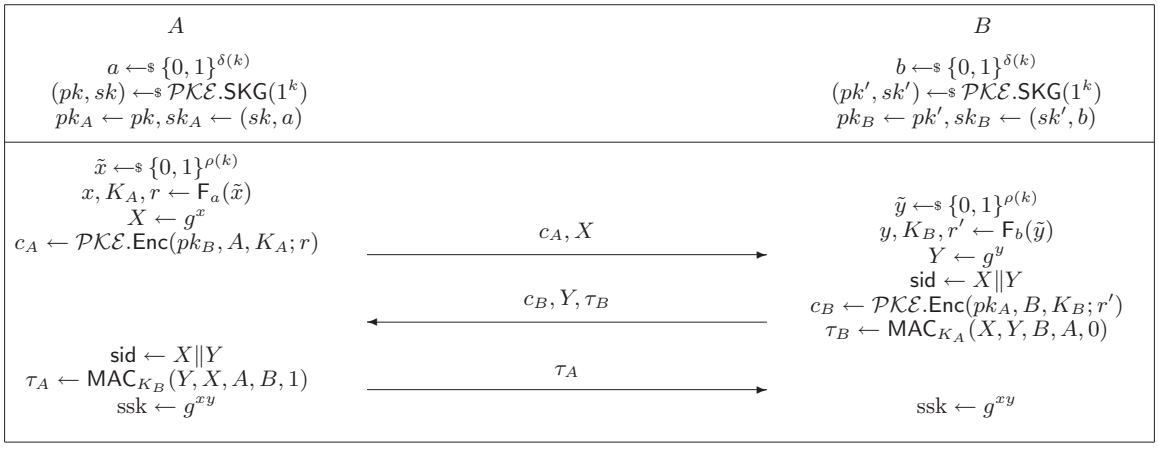

Fig. 9. The PKEDH-R Protocol.
$\mathcal{PKE}.\mathsf{Enc}(\mathrm{pk}, m; r)$ means the encryption of message m under public key pk and randomness r.

A PKE-DH Protocol Secure in Both Reset Models. Given the Reset-2 secure PKEDH-R2 protocol, we can apply the transformation in Sec. 5 to obtain a new protocol (denoted by PKEDH-R) that is secure in both Weak Corruption Reset models.

Corollary 2. *The PKEDH-R protocol in Fig. 9 is secure in (Weak Corruption) Reset-1 and Reset-2 models if $\mathcal{PKE}$ is a public key encryption scheme secure under adaptive chosen ciphertext attacks,*

$\mathcal{MAC}$ is a message authentication code scheme secure under adaptive chosen message attacks, $\mathbb{F}$ is a pseudo-random function family and a strong randomness extractor, and the DDH assumption holds in the underlying group.

7 Conclusions and Future Work

In this paper, we initiate the formal study on Authenticated Key Exchange (AKE) under bad randomness. We studied two possible situations where the randomness of an AKE protocol goes bad, and proposed two formal security models, Reset-1 and Reset-2, to capture these two bad randomness situations, respectively. We investigated the security of some widely used AKE protocols in our models, and showed that they become insecure when they use bad randomness. We then presented a way to efficiently transform a Reset-2 secure AKE protocol to a new protocol which is both Reset-1 and Reset-2 secure. Our transformation gives a modular approach to design Reset-1 and Reset-2 secure AKE protocols. We illustrated the modular approach by proposing two new AKE protocols which are strengthened versions of two widely used protocols in practice.

One of our future work is to enhance the HMQV protocol. The reset attack against HMQV given by Menezes and Ustaoglu works only when the group testings on ephemeral public keys are not performed. An interesting question is: will HMQV become secure against reset attacks if group membership testing is compulsory?

Acknowledgements

We thank Mihir Bellare for his invaluable suggestions on the modeling part of this work. We also thank the anonymous reviewers of FC 2011 for their helpful comments and suggestions. The work of H. Wang is supported in part by the Singapore National Research Foundation under Research Grant NRF-CRP2-2007-03.

References

1. Digital singature standard. National Institute of Standards and Technology, NIST FIPS PUB 186, May 1994.
2. W. Aiello, S. M. Bellovin, M. Blaze, R. Canetti, J. Ioannidis, A. D. Keromytis, and O. Reingold. Just fast keying: Key agreement in a hostile Internet. *ACM Trans. Inf. Syst. Secur.*, 7(2):242–273, 2004.
3. M. Bellare. New proofs for NMAC and HMAC: Security without collision-resistance. In *CRYPTO 2006*, pages 602–619, 2006.
4. M. Bellare, Z. Brakerski, M. Naor, T. Ristenpart, G. Segev, H. Shacham, and S. Yilek. Hedged public-key encryption: How to protect against bad randomness. In *ASIACRYPT 2009*, pages 232–249.
5. M. Bellare, R. Canetti, and H. Krawczyk. Keying hash functions for message authentication. In *CRYPTO'96*, pages 1–15, 1996.
6. M. Bellare, R. Canetti, and H. Krawczyk. Modular approach to the design and analysis of key exchange protocols. In *30th ACM STOC*, pages 419–428, 1998.
7. M. Bellare, M. Fischlin, S. Goldwasser, and S. Micali. Identification protocols secure against reset attacks. In *EUROCRYPT 2001*, pages 495–511.
8. M. Bellare and P. Rogaway. Entity authentication and key distribution. In *CRYPTO'93*, pages 232–249.
9. M. Bellare and P. Rogaway. Provably secure session key distribution — the three party case. In *28th ACM STOC*, pages 57–66.
10. S. Blake-Wilson and A. Menezes. Entity authentication and authenticated key transport protocols employing asymmetric techniques. In *Security Protocols Workshop*, pages 137–158, 1997.
11. C. Boyd, Y. Cliff, J. M. G. Nieto, and K. G. Paterson. Efficient one-round key exchange in the standard model. In *ACISP 2008*, pages 69–83. Full version available at http://eprint.iacr.org/2008/007.
12. Burton S. Kaliski Jr. An unknown key-share attack on the MQV key agreement protocol. *ACM Trans. Inf. Syst. Secur.*, 4(3):275–288, 2001.
13. R. Canetti, O. Goldreich, S. Goldwasser, and S. Micali. Resettable zero-knowledge. In *32nd ACM STOC*, pages 235–244.

14. R. Canetti and H. Krawczyk. Analysis of key-exchange protocols and their use for building secure channels. In *EUROCRYPT 2001*, pages 453–474. `http://eprint.iacr.org/2001/040/`.
15. R. Canetti and H. Krawczyk. Security analysis of IKE's signature-based key-exchange protocol. In *CRYPTO 2002*, pages 143–161. `http://eprint.iacr.org/2002/120/`.
16. O. Chevassut, P.-A. Fouque, P. Gaudry, and D. Pointcheval. Key derivation and randomness extraction. Cryptology ePrint Archive, Report 2005/061, 2005. `http://eprint.iacr.org/`.
17. A. Desai, A. Hevia, and Y. L. Yin. A practice-oriented treatment of pseudorandom number generators. In *EUROCRYPT 2002*, pages 368–383.
18. Y. Dodis. Exposure-resilient cryptography. PhD Thesis, MIT, 2000.
19. Y. Dodis, R. Gennaro, J. Håstad, H. Krawczyk, and T. Rabin. Randomness extraction and key derivation using the CBC, cascade and HMAC modes. In *CRYPTO 2004*, pages 494–510.
20. D. Eastlake, S. Crocker, and J. Schiller. *IETF RFC 1750: Randomness Recommendations for Security*, 1994.
21. A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *CRYPTO'86*, pages 186–194.
22. V. Goyal and A. Sahai. Resettably secure computation. In *EUROCRYPT 2009*, pages 54–71.
23. D. Harkins and D. Carrel. The Internet Key Exchange (IKE). RFC 2409, 1998.
24. Entity authentication mechanisms - Part 3: Entity authentication using asymmetric techniques. ISO/IEC IS 9798-3, 1993.
25. C. Kaufman. Internet Key Exchange (IKEv2) Protocol. RFC 4306, 2005.
26. H. Krawczyk. HMQV: A High-Performance Secure Diffie-Hellman Protocol. In *CRYPTO 2005*, pages 546–566.
27. H. Krawczyk. SIGMA: The 'SIGn-and-MAc' Approach to Authenticated Diffie-Hellman and Its Use in the IKE Protocols. In *CRYPTO 2003*, pages 400–425.
28. H. Krawczyk. SKEME: A versatile secure key exchange mechanism for Internet. In *NDSS*, pages 114–127, 1996.
29. B. A. LaMacchia, K. Lauter, and A. Mityagin. Stronger security of authenticated key exchange. In *Provable Security*, pages 1–16, 2007.
30. L. Law, A. Menezes, M. Qu, J. A. Solinas, and S. A. Vanstone. An efficient protocol for authenticated key agreement. *Des. Codes Cryptography*, 28(2):119–134, 2003.
31. T. Matthews. Suggestions for random number generation in software. *RSA Laboratories Bulletin # 1*, Jan. 1996.
32. A. Menezes and B. Ustaoglu. On reusing ephemeral keys in Diffie-Hellman key agreement protocols. *IJACT*, 2(2):154–158, 2010.
33. T. Okamoto. Authenticated key exchange and key encapsulation in the standard model. In *Advances in Cryptology - ASIACRYPT 2007*, pages 474–484. Full paper available at http://eprint.iacr.org/2007/473.
34. Pierre-Alain Fouque and David Pointcheval and Sébastien Zimmer. HMAC is a randomness extractor and applications to TLS. In *ASIACCS*, pages 21–32, 2008.
35. T. Ristenpart and S. Yilek. When good randomness goes bad: Virtual machine reset vulnerabilities and hedging deployed cryptography. In *Network and Distributed System Security Symposium (NDSS)*, 2010.
36. S. Yilek. Resettable public-key encryption: How to encrypt on a virtual machine. In *Topics in Cryptology - CT-RSA*, pages 41–56, 2010.